

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«Мікроелектронні та мікропроцесорні пристрої та системи»
для студентів спеціальності 141 – Електроенергетика, електротехніка
та електромеханіка всіх форм навчання
освітніх програм «Електричні та електронні апарати,
«Обладнання та інформаційні технології енергоємних виробництв»

2024

Методичні вказівки до виконання лабораторних робіт з дисципліни «Мікроелектронні та мікропроцесорні пристрої та системи» для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка освітніх програм «Електричні та електронні апарати», «Обладнання та інформаційні технології енергоємних виробництв» всіх форм навчання. /Укл.: М.О. Поляков, – Запоріжжя: Національний університет «Запорізька політехніка», 2024. – 37 с.

Укладач:

М.О. Поляков, проф., д. т. н.

Рецензент:

В.В. Василевський, доц., к. т. н.

Відповідальний
за випуск:

П.Д. Андрієнко, проф., д. т. н.

Затверджено
на засіданні кафедри
«Електричні та
електронні апарати»
Протокол № 7
від «26.12. 2023

Затверджено НМК ЕТФ
Протокол № 7
від 21.03. 2024 р.

ЗМІСТ

Вступ.....	4
Лабораторна робота № 1.....	4
Лабораторна робота № 2.....	8
Лабораторна робота № 3.....	13
Лабораторна робота № 4.....	19
Лабораторна робота № 5.....	25
Лабораторна робота № 6.....	28
Рекомендована література.....	36

ВСТУП

Метою виконання лабораторних робіт є закріплення на практиці теоретичного матеріалу з дисципліни «Мікроелектронні та мікропроцесорні пристрої та системи». Для дослідження цих систем використовується пакети середовищ програмування, трансляторів, симуляторів та мікроконтролерів.

Методичні вказівки містять опис 6 лабораторних робіт за темами програмування мовою Асемблер мікроконтролера I8051 у середовищі EdSim51, мовою С плати Ардуіно у середовищі UnoArduSim, мовою С плати Ардуіно у середовищі TINKERCAD, мовою С плати Ардуіно у середовищі Proteus, мовою С плати Ардуіно у середовищі Arduino IDE, мовою Асемблер мікроконтролерів I8051 та AVR у середовищі Proteus.

Студент зобов'язаний вивчити теоретичний матеріал [1]-[5], ознайомитись із інформаційними ресурсами, виконати практичні завдання та зробити звіт з лабораторної роботи, який буде мати основні розділи:

- назва лабораторної роботи;
- її мета;
- схеми експериментів, таблиці, графіки, розрахунки, висновки, блок-схеми алгоритмів, тексти програм на відповідних мовах програмування.

Звіт має бути складений згідно з ДСТУ 3008 – 2015 та захищений.

ЛАБОРАТОРНА РОБОТА № 1

Програмування мовою Асемблер мікроконтролеру I8051 з використанням симулятора EdSim51

Мета: вивчити команди асемблеру мікроконтролеру I8051, інтерфейс користувача симулятора EdSim51, схеми підключення периферійних пристроїв

Короткі теоретичні дані: симулятор можливо завантажити з [12]. Вікно симулятора наведено на рис. 1.1. Воно містить панелі мікроконтролера, програм, сигналів на виводах портів та вікно

периферійних пристроїв. Панель мікроконтролера дає доступ до регістрів і пам'яті програм та даних. На панелі програм розміщені наступні кнопки:

- RST – рестарт програми.
- Assm – трансляція програми.
- RUN, STEP – пуск, покрокове виконання.
- NEW, LOAD, SAVE – операції з файлами.
- COPY, Paste, EXIT – копіювати, вставити, вихід.

Панель периферійних пристроїв включає: АЦП; компаратор; 4 знаковий 7-сегментний дисплей; послідовний порт; клавіатуру; банк світлодіодів; двигун постійного струму; банк перемикачів; ЦАП; дисплей 2 рядки по 16 символів

Схему підключення цих пристроїв до портів I8051 можливо проглянути натиснувши кнопку LD на панелі периферійних пристроїв.

Хід роботи

Експеримент 1. Знайомство з інтерфейсом симулятора. Розробка простої програми.

Створити програму, що записує у резидентну пам'ять даних у комірки з адресами 41 та 42 число 1С3FH. Наступний код програми, ввести у панелі програм:

```
Load: mov R0, #41
      mov @R0, #1CH
      incR0
      mov @R0, #3FH
```

Далі натиснути кнопку Assm. Далі натиснути кнопку STEP та спостерігати зміни у панелях.

Завдання 1. Прокоментувати зміст комірок пам'яті програм.

Завдання 2. Записати зміни, що відбулися після кожного кроку програми.

Завдання 3. Зберегти програму у файлі.

Завдання 4. Розробити та продемонструвати програму, яка обчислює логічну функцію. Вираз функції та прив'язку її змінних до пінів мікроконтролера визначає викладач.

The screenshot displays the EdSim51 simulator interface, which is divided into several functional areas:

- Assembly Editor (Top):** Contains assembly code with instructions like `Load: mov R0, #41`, `mov R0, #1CH`, `inc R0`, and `mov R0, #3FH`. It includes a menu bar (RST, Asm, Run, New, Load, Save, Copy, Paste) and a status bar.
- Registers and Status (Middle Left):** Shows the status of various registers (R0-R7, TH0, TL0, W/O, R/O, TXD, RXD, SC0N, pins bits) and their current values. The PC register is highlighted with the value 8051.
- Memory View (Middle Right):** Displays a table of memory addresses (0 to 70) and their corresponding values (0x00 to 0xFF).
- Hardware Components (Bottom):** Includes a digital display showing '88.88', a keypad, a motor control section with 'MAX' and 'MIN' buttons, and a status indicator for 'Motor Enabled'.

Copyright ©2005-2022 James Rogers

Рисунок 1.1 – Вікно симулятора EdSim51

Експеримент 2. Розробка програми що керує периферійним пристроєм.

Створити програму що виконує послідовну індикацію цифр від 0 до 9 на правий 7-сегментний індикатор у панелі периферійних пристроїв. Ввести та виконати програму з кодом наведеним на рис. 1.2.

```

mov 30h, #11000000b
mov 31h, #11111001b
mov 32h, #10100100b
mov 33h, #10110000b
mov 34h, #10011001b
mov 35h, #10010010b
mov 36h, #10000010b
mov 37h, #11111000b
mov 38h, #10000000b
mov 39h, #10010000b
mov 3ah, #0
clr p3.4
clr p3.3
start: mov r0, #30h
loop:  mov a, @r0
       jz start
       mov p1, a
       inc r0
       jmp loop

```

Рисунок 1.2 Програма індикації

Завдання 1. Розробити блок-схему алгоритму програми індикації.

Завдання 2. Модернізувати програму, щоб вона відображала усі шістнадцятирічні цифри.

Завдання 3. Модернізувати програму, щоб вона відображала цифри на іншому індикаторі.

Завдання 4. Змінити час індикації однієї цифри.

Завдання 5. Навести схему підключення індикаторів до мікроконтролера

Контрольні запитання:

1. Призначення панелей симулятора.
2. За допомогою яких пінів мікроконтролера керується двигун постійного струму. Навести схему підключення.
3. Що треба зробити, щоб шістнадцятиричну цифру «С» вивести на певний 7-сегментний індикатор.
4. Як перевести код з пам'яті програм на мову асемблер.

ЛАБОРАТОРНА РОБОТА № 2**Програмування мовою С плати Ардуіно у середовищі
UnoArduSim**

Мета: Навчитись програмувати мікроконтролерні системи на базі плат Ардуіно (Arduino) мовою C/C++ у середовищі UnoArduSim

Короткі теоретичні дані: Плати Ардуіно це поширений, доступний засіб для управління електромеханічним обладнанням за допомогою мікроконтролерів. Вони містять контакти для під'єднання давачів та(або) виконавчих механізмів. У простішому варіанті це світлодіоди, вимикачі, Ці та інші електричні та радіо компоненти монтують на спеціальних платах. Зв'язок плати Arduino із системою програмування виконується за допомогою USB кабелю. Програмне забезпечення UnoArduSim, Arduino IDE для програмування мікропроцесорів плат Arduino мовою С є у вільному доступі в мережі Інтернет.

Рекомендована література [9], довідка про мову програмування [10].

Для мікроконтролерів широкого призначення існують програмні середовища, що здійснюють програмування, налагодження програми, проектування апаратних компонент тощо.

Так, наприклад, для програмування контролерів Arduino використовується середовище розробки Arduino, яке складається з безкоштовної програмної оболонки (IDE) для написання програм, їх компіляції та програмування апаратури.

Мова програмування Arduino є стандартною мовою C++ з особливостями, які полегшують непідготовленим користувачам написання працюючої програми.

Програми для Arduino називають скетчі (від англ. Sketch), які складаються в текстовому редакторі. Після компіляції програма зберігається в файлах з розширенням `.ino`. Перед компіляцією ці файли обробляються препроцесором Arduino, у ході чого здійснюється виявлення помилок у разі наявності.

Також існує можливість створювати і підключати до проекту стандартні файли C++. Обов'язкову в C++ функцію `main ()` препроцесор Arduino створює сам, вставляючи туди необхідні дії.

Програміст повинен написати дві обов'язкові для Arduino функції `setup ()` і `loop ()`. Перша викликається одноразово при старті, друга виконується в нескінченному циклі. В тексті своєї програми (скетчу) програміст може використовувати стандартні бібліотеки.

Менеджер проекту Arduino IDE має нестандартний механізм додавання бібліотек. Бібліотеки у вигляді вихідних текстів на стандартному C++ додаються в спеціальну папку в робочому каталозі IDE. При цьому назва бібліотеки буде додано до списку бібліотек в меню IDE. Програміст зазначає потрібні бібліотеки і вони вносяться до списку компіляції.

Arduino IDE не пропонує ніяких налаштувань компілятора і мінімізує інші настройки, що спрощує початок роботи для новачків і зменшує ризик виникнення помилок.

Загальний вид екрану програми Arduino IDE наведено на рис. 2.1.

Загальний вид екрану програми UnoArduSim наведено на рис. 2.2.

Ця програма містить симулятори плати Arduino та моделі об'єктів керування, таких як світлодіоди, двигуни, індикатори тощо. Тобто запустив програму на виконання ми бачимо анімацію обертання двигуна, миготіння світлодіода та таке інше. Але за допомогою програми UnoArduSim не можливо завантажити програму у реальну плату Arduino.

Найпростіша Arduino-програма складається з двох функцій:

- `setup ()`: функція викликається одноразово при старті мікроконтролера.

- `loop ()`: функція викликається після `setup ()` в нескінченному циклі весь час роботи мікроконтролера.

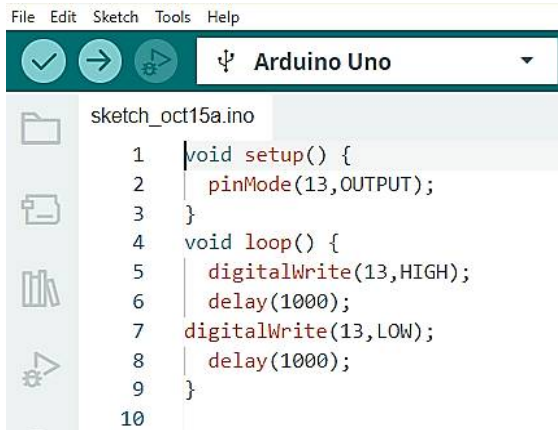


Рисунок 2.1 – Загальний вид екрану програми Arduino IDE

На рис. 2.1 та 2.3 наведено повний текст однієї з найпростіших програм (скетчу) миготіння світлодіодом, підключеного до виходу 13 Arduino, з періодом 1 секунди, де для управління цифровим виходом використовується функція `digitalWrite()`. Ця функція має два аргументи: номер виходу, на який виводиться дискретний сигнал, і значення сигналів HIGH (високий рівень) або LOW (низький рівень).

Всі використовувані в прикладі функції є бібліотечними. У комплекті Arduino IDE та UnoArduSim є велика кількість прикладів програм.

При виконанні з реальною платою Arduino скетчу рис. 2.1 спостерігати миготіння вбудованого світлодіоду. Зовнішній світлодіод та послідовний з ним резистор приєднати з одного боку до пін 13, а з іншого – до вивода GND (Загальний). Опір резистора розрахуємо за формулою:

$$R = (E - U_d) / I_d$$

де E – напруга живлення; U_d – напруга на діоді; I_d – струм через діод.

У випадку, коли відсутні реальна плата Ардуіно, зовнішні радіоелементи та пристрої, використовується програма-симулятор UnoArduSim [11].

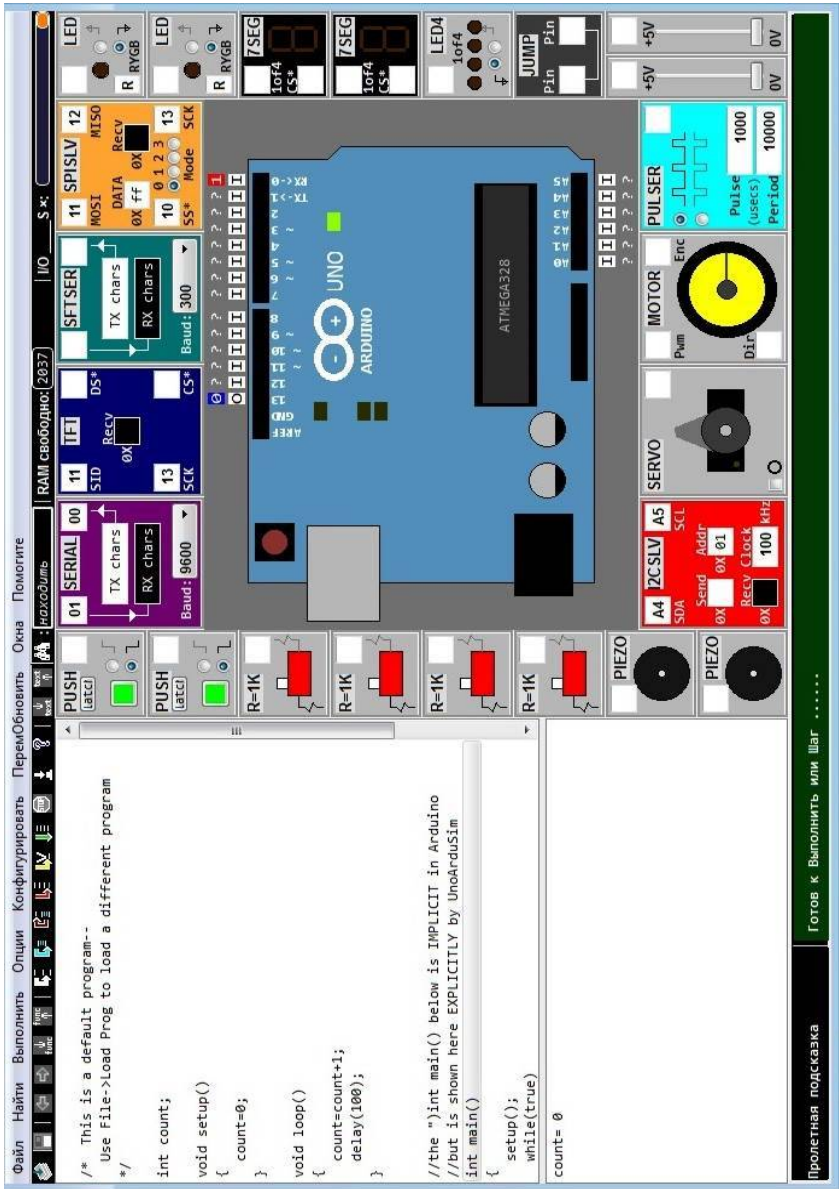


Рисунок 2.2. – Экран симулятора UnoArduSim

```

void setup () {
  pinMode (13, OUTPUT); // Визначення вивода 13 як вихід
}
void loop () {
  digitalWrite (13, HIGH); // Включення _виводу 13, параметр HIGH
  // - високого признак високого логічного рівня
  delay (1000); // Цикл затримки на 1000 мс – 1 секунда
  digitalWrite (13, LOW); // Виключення вивода 13, параметр LOW
  delay (1000); // Цикл затримки на 1000 мс – 1 секунда }

```

Рисунок 2.3 – Текст програми (скетчу) миготіння світло діодом

Хід роботи

Експеримент 1. Миготіння світлодіодом.

Завдання 1

1. Відкрити симулятор UnoArduSim. Перейти у режим редагування скетчу (подвійний клік на білому полі скетчу).
2. У вікні редагування набрати текст скетчу миготінні вбудованим світлодіодом 13.
3. Скопіювати скетч (перекласти з мови програмування C+ на мову яку розуміє мікроконтролер) – натиснути ^R.
4. Завантажити скетч до мікроконтролеру -натиснути ^U.
5. Виконати скетч - натиснути функціональну клавішу f9 на клавіатурі комп'ютеру. Спостерігати миготіння: 1 секунду «горить», 1 секунду триває пауза. На піні 13 також змінюється колір.

Завдання 2. Змінити частоту миготіння.

Завдання 3. Додати до схеми зовнішній світлодіод.

Експеримент 2. Керування двигуном постійного струму.

Завдання 1. У середовищі симулятора UnoArduSim ввести та виконати скетч, що керує обертанням двигуна із заданою швидкістю.

Завдання 2. Змінити напрям обертання двигуна.

Завдання 3. За допомогою моделі потенціометра задавати значення змінної x . Якщо $x \geq 500$ то двигун повинен обертатись «вправо», інакше – «вліво»

Завдання 4. Скласти програму для керування швидкості двигуна постійного струму за допомогою моделі потенціометра.

Контрольні питання

1. Системні характеристики плат Arduino.
2. Інтерфейс середовища UnoArduSim.
3. Структура програми керування.
4. Як запрограмувати лінію порта як вхід або вихід.
5. Що таке ШІМ.
6. Параметри АЦП плати Arduino

ЛАБОРАТОРНА РОБОТА № 3

Програмування мовою C плати Ардуіно у середовищі TINKERCAD

Мета Навчитися програмувати мовою програмування C плати Ардуіно у середовищі TINKERCAD.

Короткі теоретичні дані: Tinkercad — це безкоштовна онлайн-програма для 3D-моделювання, яка працює у веб-браузері [13].

Розділ Tinkercad's Circuits — це браузерний симулятор електронних схем, який підтримує мікроконтролери Arduino Uno, плати Micro:bit або мікросхеми ATtiny. Код може бути створений за допомогою текстового коду або графічних CodeBlocks – фрагментів коду, які можна легко впорядкувати за допомогою миші.

Tinkercad поставляється з вбудованими бібліотеками для популярних компонентів, включаючи бібліотеки дисплеїв Adafruit Neopixel, Arduino Servo та I²C. Однак користувацькі бібліотеки не можна вибрати або завантажити. Симулятор також підтримує аналогові компоненти, які повністю змодельовані.

Tinkercad пропонує розширені функції, такі як багатоплатне моделювання та складні аналогові схеми.

Інтерфейс користувача містить головне меню та панелі. Додаток відображає всі компоненти, які ви можете розмістити у своєму дизайні, на панелі в правій частині інтерфейсу. Перетягніть частину в центральну область інтерфейсу користувача, щоб додати її до

дизайну. Ви можете обертати компоненти за допомогою рядка головного меню програми або натискання кнопки **g** на клавіатурі. Рядок головного меню також дозволяє видаляти деталі, скасовувати та повторювати дії, додавати примітки до дизайну, а також змінювати кольори проводів і типи роз'ємів.

Три кнопки у верхньому правому куті програми дозволяють перемикатися між різними поданнями. Вікно симулятора є першим видом, і воно відкрите за замовчуванням. У другому поданні міститься автоматично згенерована принципова схема вашого проекту, а в третьому поданні можна експортувати позначку порядку позначення, у якій перелічено всі компоненти, додані до схеми.

Почніть свій проєкт, розмістивши плату Arduino UNO та макетну плату в головній робочій області програми в центрі інтерфейсу користувача. Ви можете скористатися рядком пошуку на панелі компонентів у правій частині програми, щоб швидко знайти деталі. Потім перетягніть компоненти в проєкт. Ви можете показувати додаткові параметри для коригування значень складників одинарним клацанням лівою кнопкою миші на частині. Розмістіть компоненти електричної схеми на невеликій макетній платі. Потім під'єднайте компоненти до плати Arduino, Приклад електричної схеми наведено на рис. 3.1.

Для перемикання між редактором блоків і звичайним текстовим редактором вихідного коду використовувати виділену кнопку. Вставте код вашого проєкту в текстовий редактор, щоб запрограмувати Arduino. Потім закрийте панель коду. Коли все буде готово, натисніть кнопку симуляції запуску поруч із кнопкою коду. Має розпочатися симуляція.

Поруч із кнопкою запуску симуляції ви можете знайти ще одну кнопку, щоб експортувати та поділитися своїм дизайном. Схеми Tinkercad можуть експортувати ваш проєкт у вигляді файлу PNG або файлу проєкту плати EAGLE/Fusion360. За допомогою цього файлу проєкту плати ви можете провести свій експеримент і перетворити його на проєкт друкованої плати.

Керування двигуном постійного струму – одна з поширених завдань моделювання в Tinkercad. Електрична схема керування містить мікросхеми драйверів L293 і L293D та логічних елементів I-H174HC00.

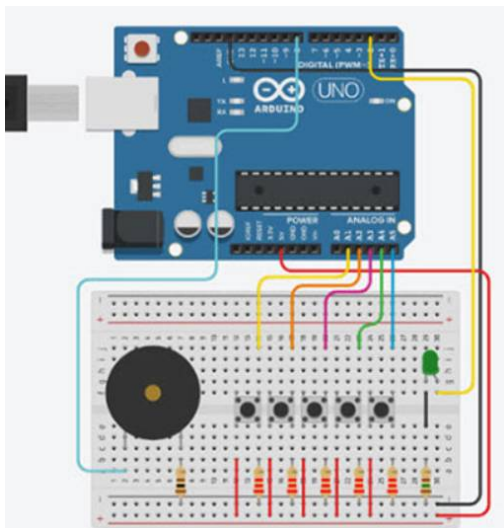


Рисунок 3.1 – Приклад електричної схеми

Моделі L293 та L293D мають TTL-сумісні входи та високовольтні виходи для керування індуктивним навантаженням. Токові виходи можуть досягати 2 А при використанні L293.

Кожен вихід є повною схемою збудження з тотемним полюсом, що складається зі стоку транзистора Дарлінгтона і витoku транзистора псевдо Дарлінгтона.

Схема та конфігурація пінов драйвера двигуна наведена на рис. 3.2 та рис. 3.3. Драйвери включаються попарно: драйвери 1 та 2 включаються за допомогою 1,2EN, а драйвери 3 та 4 включаються за допомогою 3,4EN.

Коли вхід дозволу має високий рівень, відповідні драйвери включені, а їх виходи активні і знаходяться у фазі з входами. Коли на вході дозволу низький рівень, ці драйвери відключені, а їх виходи вимкнені та перебувають у стані високого імпедансу.

При правильному введенні даних кожна пара драйверів утворює реверсивний привід з повним Н-подібним (або мостовим) приводом, що підходить для соленоїдів або двигунів. Детальнішу інформацію про ці мікросхеми можливо отримати з datasheet, які є на сайті [datasheet \[15\]](#).

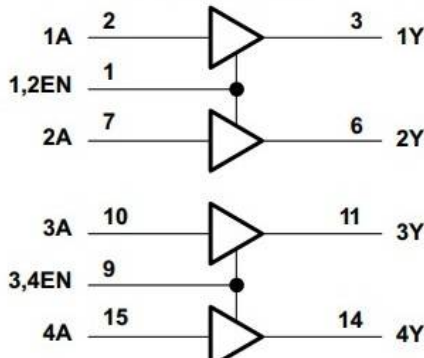


Рисунок 3.2 – Схема драйвера двигуна

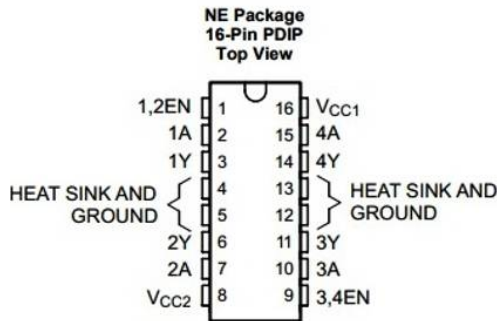


Рисунок 3.3 – Конфігурація пінов драйвера двигуна

Хід роботи

Експеримент 1. Робота з осцилографом

Завдання 1. У середовищі Tinkercad створити електричну схему, ввести код програми що наведено на рис. 3.4. Знати та виправити помилку у коді. Запустити виконання проекту. Визначити період ШІМ.

Завдання 2. Модернізувати програму таким чином, щоб канали 3, 11 працювали у протифазі та були навантажені світлодіодами.

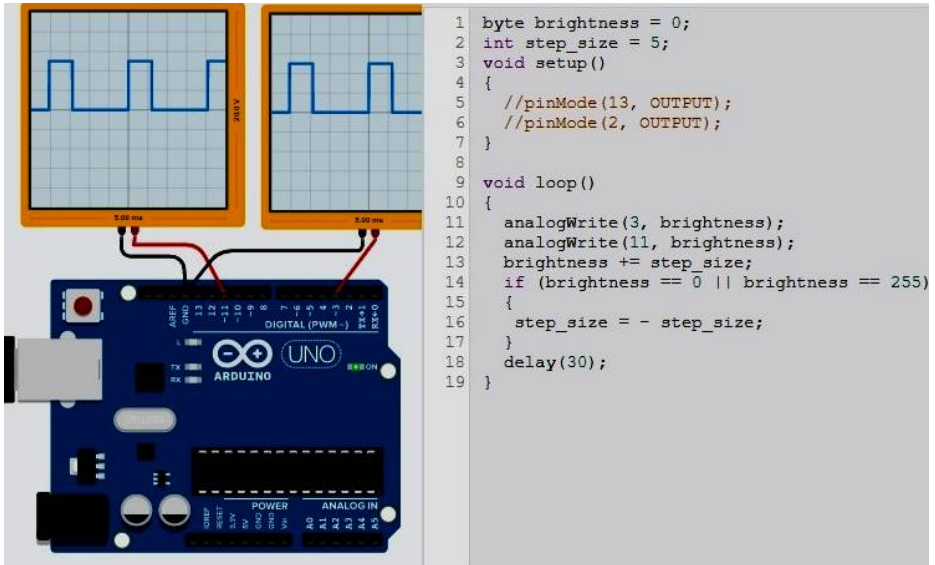


Рисунок 3.4 – Електрична схема експерименту

Експеримент 2. Smart Home's Kitchen.

Завдання 1. На сайті Tinkercad знайти проєкт Smart Home's Kitchen, завантажити його, запустити на виконання, визначити функції керування, зробити опис алгоритму керування.

Експеримент 3. Керування двигуном постійного струму.

Завдання 1. Під'єднати двигун до драйвера, як наведено на рис. 3.5.

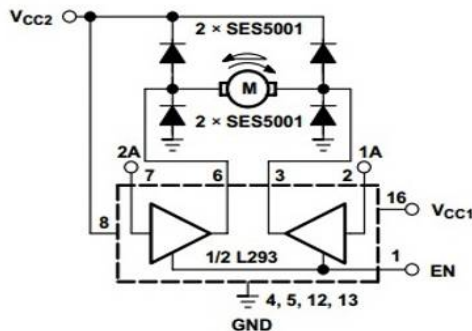


Рисунок 3.5 – Під'єднання двигуна до драйвера

Завдання 2. У середовищі Tinkercad створити електричну схему, що наведена на рис. 3.6 та містить плату Arduino UNO, макетну плату, драйвер L293D, мікросхему I-HI (NAND)-74HC00, кнопку «направлення обертання», потенціометр «швидкість обертання» та резистор 1 кОм.

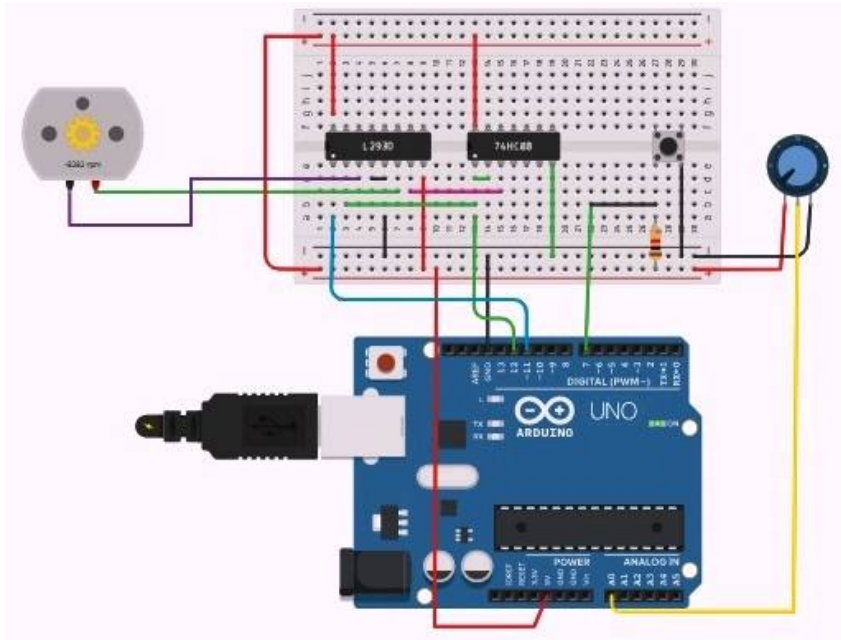


Рисунок 3.6 – Під'єднання двигуна до драйвера

Завдання 3. Розробити, ввести та виконати скетч, що керує обертанням двигуна зі заданою швидкістю та напрямом.

Контрольні питання

1. Інтерфейс середовища Tinkercad.
2. Як у середовищі створити та роздрукувати електричну схему, ввести код програми.
3. Що таке ШІМ.
4. Поясніть як працює оператор if у коді експерименту1.
5. Як працює драйвер двигуна.
6. Як запрограмувати лінію порту як вхід або вихід.

ЛАБОРАТОРНА РОБОТА № 4

Програмування мовою С плати Ардуіно у середовищі Proteus

Мета: вивчити інтерфейс середовища Proteus та процес програмування мовою С плати Ардуіно у середовищі Proteus.

Короткі теоретичні дані:

Proteus VSM – це програма-симулятор радіотехнічних схем та мікроконтролерних пристроїв, що підтримує мікроконтролери PIC, 8051, AVR, HC11, ARM7/LPC2000 та інші [14].

Proteus VSM складається з двох основних модулів:

- ISIS – графічний редактор принципових схем, який використовується для вводу розроблених проектів з подальшою імітацією і передачею для розробки друкованих плат в ARES. Після налагодження пристрою можна відразу розвести друковану плату в ARES, який підтримує авто розміщення і трасування за вже існуючою схемою.

- ARES – графічний редактор друкованих плат з вбудованим менеджером бібліотек і трасувальником ELECTRA, автоматичною розстановкою компонентів на друкованій платі.

Головний екран програми наведено на рис. 4.1. Основні пункти меню Proteus ISIS: File, View, Edit, Library, Tools, Design, Graph, Source, Debug, Template, System, Help.

Для початкового знайомства з програмою Proteus корисно розглянути приклади натиснув Open Sample. При створенні нового проєкту (File – > New Project) треба вибрати назву нового проєкту та локацію для збереження на комп'ютері. Наступний крок – вибір розміру полотна для схеми будемо обирати тип DEFAULT – за замовчуванням. У наступному вікні можна обрати налаштування для створення проєкту прошивки мікроконтролера (оберемо – без прошивки). А далі закінчуємо процес попереднього налаштування проєкту. Після чого відкриється підсумкове вікно створення проєкту. Приклад відкритого проєкту наведено на рисунку 4.2.

Інтерфейс середовища ISIS містить список компонентів, що були використані для побудови даної схеми, стрілки повороту та

стрілки напрямів у меню посередині що дозволяють більш точно рухати елемент на схемі.

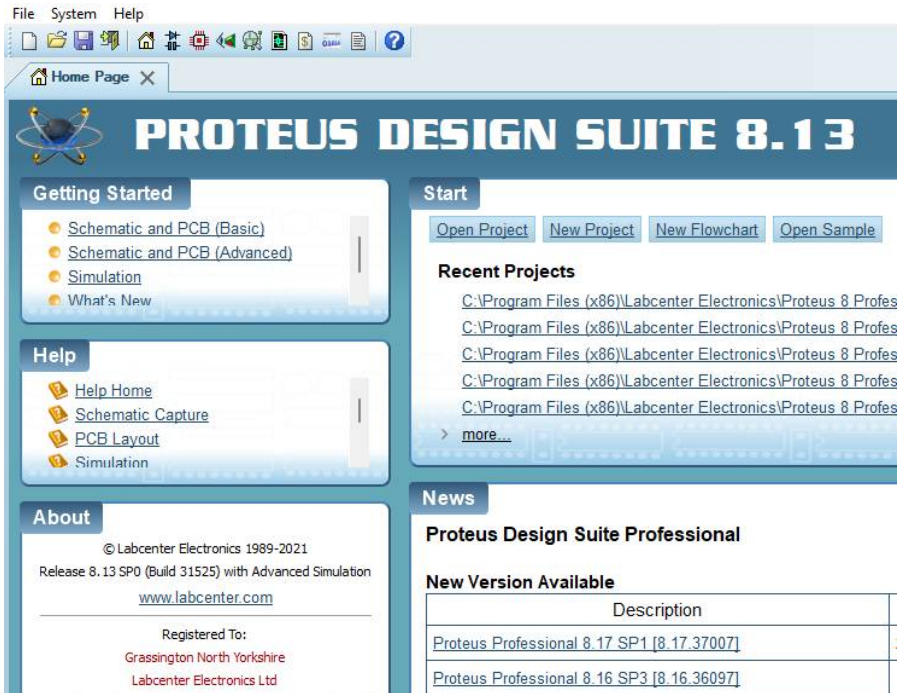


Рисунок 4.1 – Головний екран програми Proteus

Для запуску схеми використовується меню, що знаходиться внизу вікна ISIS. Щоб розпочати виконання програми треба натиснути на синій трикутник, а щоб завершити – на квадрат.

Щоб додати будь-яку модель на схему потрібно слідувати наступним крокам:

1. У головному вікні схематичного представлення мікросхеми треба натиснути правою кнопкою миші по вільній ділянці полотна. Відкриється невелике меню, з якого треба буде обрати перший пункт – Place (помістити).

2. У наступному меню потрібно обрати, який саме елемент ми хочемо помістити на схему. Обираємо варіант Component.

Рисунок 4.2 – Приклад відкритого проекту.

3. Третє меню дозволяє обрати недавно використані елементи. Ми ж оберемо останній пункт меню – From libraries (з бібліотек), щоб самостійно обрати потрібний елемент з бібліотеки.

4. У вікні, що з'явилося, впишемо у строку пошуку в лівому верхньому кутку вікна ключові слова: motor. Після чого буде знайдено усі елементи, що використовують це слово у своїй назві.

5. Обравши потрібний елемент натиснемо кнопку ОК та розмістимо елемент на схемі клацнувши по ній лівою кнопкою миші.

Щоб додати осцилограф та вольтметр потрібно переключитися у режим Instruments в меню. Відкриється панель інструментів, з якої можна обрати потрібний нам. Під'єднаємо осцилограф до схеми. Для цього треба кликнути лівою кнопкою миші по одному з виходів осцилографа, має з'явитися лінія, що слідує за курсором миші. Під'єднаємо інший кінець цього дроту до ділянки кола, яку нам треба прослухати. Для цього клацнемо ще раз по ділянці, до якої потрібно під'єднатися. З'явиться лінія між осцилографом та ділянкою кола.

Під'єднаємо інший кінець цього дроту до ділянки кола, яку нам треба прослухати. Для цього клацнемо ще раз по ділянці, до якої потрібно під'єднатися. З'явиться лінія між осцилографом та ділянкою кола.

Щоб відкрити вікно осцилографа, треба запустити схему. Для цього треба натиснути на синій трикутник в нижній частині екрану. Далі треба у пункті меню Debug на верхній панелі у головному вікні програми обрати зі списку компонентів системи осцилограф і натиснути на його назву. Одразу ж відкриється вікно осцилографа з даними.

Більш детальна інформація по програмі Proteus є в посібнику [4].

Хід роботи

1. Обрати приклад – проєкт Arduino & Common Cathode RGB LED.
2. Проаналізувати схему (рис. 4.2) та код (рис. 4.3) програми що керує.
3. Запустити виконання проєкту та спостерігати на осцилографі процес зміни параметрів ШІМ (приклад наведено на рис. 4.4).
4. Визначити період ШІМ.
5. Створити проєкт у якому до плати Arduino під'єднано світлодіоди. Як варіант можливо модифікувати існуючий проєкт.

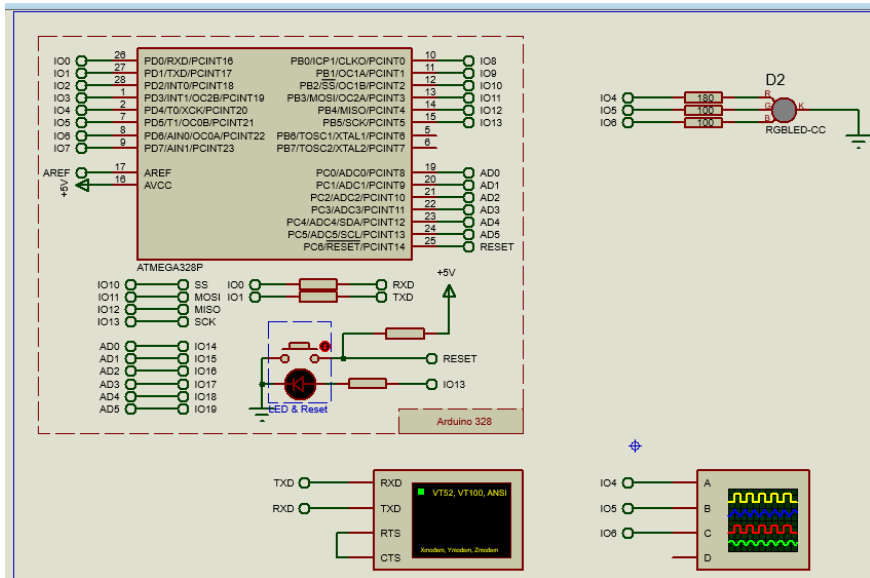


Рисунок 4.2 – Схема проекту Arduino & Common Cathode RGB LED.

```

*main.ino
1  int REDPin = 4; // RED pin of the LED to PWM pin 4
2  int GREENPin = 5; // GREEN pin of the LED to PWM pin 5
3  int BLUEPin = 6; // BLUE pin of the LED to PWM pin 6
4
5  int brightness = 0; // LED brightness
6  int increment = 5; // brightness increment
7
8  void setup()
9  { pinMode(REDPin, OUTPUT);
10   pinMode(GREENPin, OUTPUT);
11   pinMode(BLUEPin, OUTPUT);
12   Serial.begin(9600);
13 }
14
15 void loop()
16 { brightness = brightness + increment; // increment brightness for next loop iteration
17   if (brightness <= 0 || brightness >= 255) // reverse the direction of the fading
18   { increment = -increment;
19   }
20   brightness = constrain(brightness, 0, 255);
21
22   analogWrite(REDPin, brightness);
23   analogWrite(GREENPin, brightness);
24   analogWrite(BLUEPin, brightness);
25   delay(20); // wait for 20 milliseconds to see the dimming effect
26 }

```

Рисунок 4.3 – Код програми що керує RGB LED.

Контрольні питання

1. Особливості інтегрованого середовища розробки Proteus ISIS.
2. Основні пункти меню Proteus ISIS.
3. Створення проекту в Proteus ISIS.
4. Як визначити опір резистору, що під'єднаний послідовно зі світлодіодом.

ЛАБОРАТОРНА РОБОТА № 5

Програмування мовою C плати Ардуіно у середовищі Arduino IDE

Мета: вивчити процес програмування мовою C плати Ардуіно у середовищі Arduino IDE для керування реальними об'єктами.

Короткі теоретичні дані: Arduino IDE це поширений пакет для програмування мовою C мікроконтролерних плат, в тому числі плати Ардуіно. Програма керування створена у цьому пакеті завантажується у реальну плату, до якою під'єднані реальні радіоелементи або/та пристрої. Загальний вид екрану програми Arduino IDE наведено на рис. 2.1.

Програма Arduino IDE, відповідні драйвера плат та потрібні бібліотеки повинні бути встановлені на комп'ютері робочого міста. Крім того плата Arduino повинна бути під'єднана до цього комп'ютера за допомогою USB кабелю. Процес створення програми керування у середовищі Arduino IDE практично не відрізняється від процесу в симуляторі. Виконанню розробленої програми керування передуює етап складання електричною схеми експерименту. Розглянемо цей процес на прикладі схеми наведеної на рис. 5.1.

На макетній платі стенду зібрати схему: шина GND – резистор 220 Ом – світлодіод – шина +5 В. під'єднати стенд кабелем USB до комп'ютеру. Спостерігати, що діод «горить» тільки тоді, коли анод діода (вивід більшої довжини) під'єднаний до шини +5 В. Якщо під'єднати діод у протилежному напрямі то діод не пропускає струм і не «горить». На макетній платі стенду зібрати схему: шина GND – резистор 220 Ом – світлодіод (анодом – виводом більшої довжини

під'єднати до Ардуіно) – пін 13 плати Ардуіно. Перейти у середовище Arduino IDE.

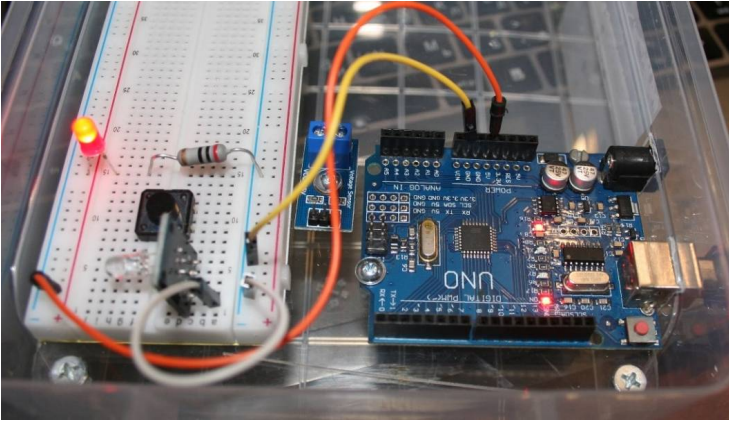


Рисунок 5.1 – Макет електричної схеми

Хід роботи

4.1. У середовищі **Arduino IDE** набрати текст скетчу що наведено на рис. 5.2.

```

1 void setup() {
2   pinMode(13,OUTPUT);
3 }
4 void loop() {
5   digitalWrite(13,HIGH);
6   delay(1000);
7   digitalWrite(13,LOW);
8   delay(1000);
9 }
10

```

Рисунок 5.2 – Текст скетчу

4.2. У меню **Tools/Board/Arduino AVR Boards** обрати **Arduino Uno** (на деяких стендах- **Mega 2560**).

4.3. У меню **Tools/Ports/Serial Ports** обрати мишкою доступний порт(стенд повинен бути під'єднаний до комп'ютеру).

4.4. Натиснути (компіляція), а потім, якщо немає помилок натиснути (завантаження та запуск) (див. рис. 5.3).

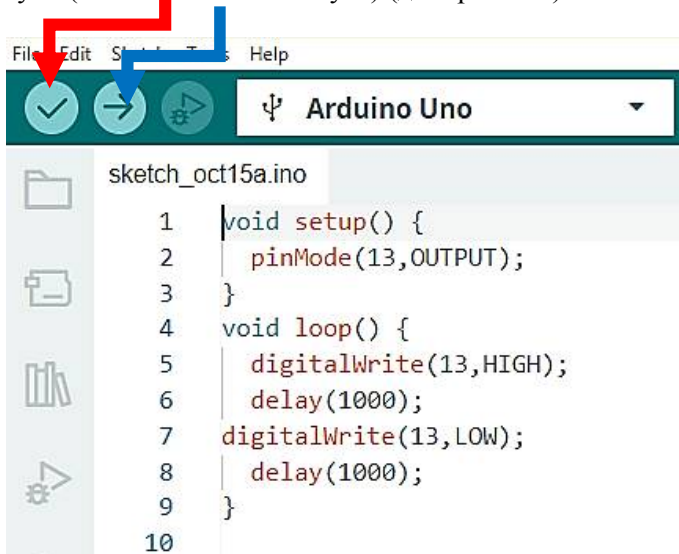


Рисунок 5.3 – Дії при компіляції та завантаженні програми.

Спостерігати миготіння як вбудованого так і зовнішнього світлодіодів.

4.5. Відкоригувати скетч з метою зміни тривалості «горіння», потім під'єднати світлодіод до іншого піну, та відповідно змінити скетч.

4.6. Перейти до експериментів, що виконуються тільки для вашого стенду.

Контрольні питання

1. Особливості інтегрованого середовища Arduino IDE.
2. Основні пункти меню Arduino IDE.
3. Створення проекту в Arduino IDE.

4. Як визначити опір резистору, що під'єднаний послідовно зі світлодіодом.

ЛАБОРАТОРНА РОБОТА № 6

Програмування мовою Асемблер мікроконтролерів I8051 та AVR у середовищі Proteus

Мета: вивчити команди асемблеру, навчитися керувати виведенням на дисплей, організовувати цикли затримки при виводі інформації на порти та візуалізувати отримані результати на світлодіодах у програмі Proteus VSM.

Короткі теоретичні дані: Інтерфейс середовища Proteus VSM розглянуто у попередніх лабораторних роботах.

Хід роботи

- 1.Завантажити вбудований приклад 8051 LCD Display Driver. Електрична схема прикладу наведена на рис. 6.1.

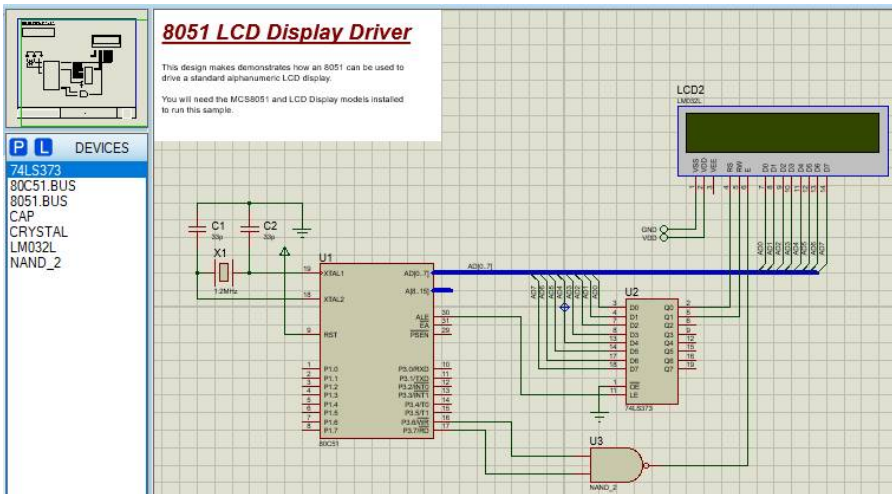


Рисунок 6.1 – Електрична схема прикладу 8051 LCD Display Driver

2. За допомогою піктограми Bill of Materials визначити тип елементів електричної схеми. Виконати пошук у інтернеті описів елементів U1-U4. Визначити з описів функції цих елементів та призначення їх пінів.

3. На вкладці Source Code ознайомитись зі структурою та командами програми що наведена нижче:

```
; LCD Display Driver Demo.
```

```
; Timing code assumes 1.2MHz Clock
```

```
#include <bitdef.h>
```

```
#include <io80C51BH.h>
```

```
;LCD Registers addresses
```

```
LCD_CMD_WR equ 0
```

```
LCD_DATA_WR equ 1
```

```
LCD_BUSY_RD equ 2
```

```
LCD_DATA_RD equ 3
```

```
;LCD Commands
```

```
LCD_CLS equ 1
```

```
LCD_HOME equ 2
```

```
LCD_SETMODE equ 4
```

```
LCD_SETVISIBLE equ 8
```

```
LCD_SHIFT equ 16
```

```
LCD_SETFUNCTION equ 32
```

```
LCD_SETCGADDR equ 64
```

```
LCD_SETDDADDR equ 128
```

```
;Reset vector
```

```
org 0000h
```

```
jmp start
```

```
;Start of the program
```

```
org 0100h
```

```
string1a:db '!! A M A Z I N G !!'
```

```
db 0
```

```
string1b:db '!! A M A Z I N G !!'
```

```
db 0
```

```
string2:db 'A virtual LM032L...'
```

```
db 0
```

```
string3:db 'driven by a virtual'
```

```

db 0
string4:db ' 8051 processor!'
db 0
start:mov A,#038h
call wrcmd
loop:mov A,#LCD_SETVISIBLE+6 ;Make the display & blink visible:
call wrcmd
mov R7,#2
loop2:
mov DPTR,#string1a
call wrstr
mov DPTR,#200
call wtms
mov A,#LCD_CLS;Clear screen
call wrcmd
mov DPTR,#string1b
call wrstr
mov DPTR,#200
call wtms
mov A,#LCD_CLS;Clear screen
call wrcmd
djnz R7,loop2
mov DPTR,#string1a
call wrstr
mov DPTR,#400
call wtms
mov A,#LCD_SETDDADDR+64
call wrcmd
mov DPTR,#string2
call wrslow
mov DPTR,#200
call wtms
mov A,#LCD_CLS;Clear screen
call wrcmd
mov DPTR,#string3

```

```

call wrslow
mov A,#LCD_SETDDADDR+64
call wrcmd
mov DPTR,#string4
call wrslow
mov A,#LCD_SETVISIBLE+7;Show the blink cursor as well.
call wrcmd
mov DPTR,#2000
call wtms
mov A,#LCD_CLS;Clear screen
call wrcmd
jmp loop
;Sub routine to write null terminated string at DPTR in program ram.
wrstr:mov R0,#LCD_DATA_WR
wrstr1:clr A
movc A,@A+DPTR
jz wrstr2
movx @R0,A
call wtbusy
inc DPTR
push DPL
push DPH
pop DPH
pop DPL
jmp wrstr1
wrstr2:ret
;Sub routine to write null terminated string at DPTR in program ram.

```

Slowly

```

wrslow:mov R0,#LCD_DATA_WR
wrslw1:clr A
movc A,@A+DPTR
jz wrslw2
movx @R0,A
call wtbusy
inc DPTR

```

```

push DPL
push DPH
mov DPTR,#100
call wtms
pop DPH
pop DPL
jmp wrslw1
wrslw2:ret
;Sub routine to write command:
wrcmd:mov R0,#LCD_CMD_WR
movx @R0,A
jmp wtbusy
;Sub routine to write character:
wrchar:mov R0,#LCD_DATA_WR
movx @R0,A
;Subroutine to wait for busy clear
wtbusy: mov R1,#LCD_BUSY_RD
movx A,@r1
jb ACC.7,wtbusy
ret
;Wait for number of seconds in A
wtsec:push ACC
call wtms
pop ACC
dec A
jnz wtsec
ret
;Wait for number of milliseconds in DPTR
wtms: xrl DPL,#0FFh;Can't do DEC DPTR, so do the loop by forming
2's complement
xrl DPH,#0FFh;and incrementing instead.
inc DPTR
wtms1:mov TL0,#09Ch;100 ticks before overflow = 1ms at 1.2MHz
Clock
mov TH0,#0FFh

```

```

mov TMOD,#1;Timer 0 mode 1
setb TCON.4;Timer 0 runs
wtms2:jnb TCON.5,wtms2
clr TCON.4;Timer 0 stops
clr TCON.5
inc DPTR
mov A,DPL
orl A,DPH
jnz wtms1
ret
END

```

4. Ввести та виконати цю програму у середовищі Proteus. Модифікувати програму у частини інформації яка виводиться на дисплей та у частини швидкості зміни тексту на екрані дисплею.

5. За допомогою вікон меню Debug/8051 CPU/Registers та інших, отримати доступ до спостереження за змінами у цих елементах. Далі за допомогою вікон меню Debug/Start VSM Debugging/Step Over Source Line та Step Into Source Line виконати програму в покроковому режимі.

6. Побудувати схему наведену на рис. 6.2.

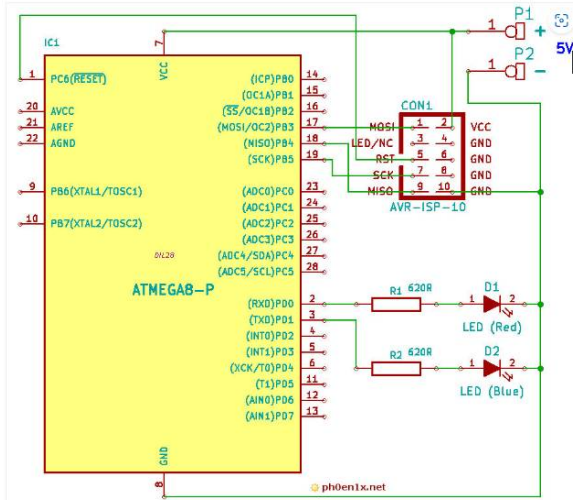


Рисунок 6.2 – Електрична схема що миготить двома світлодіодами

7. Ввести та виконати у середовищі Proteus у звичайному та покроковому режимах програму що наведено на рис. 6.3 та 6.4.

```

1  ; Светодиодная мигалка на микроконтроллере ATmega8
2  ; https://ph0en1x.net
3
4  .INCLUDEPATH "/usr/share/avra/" ; путь для подгрузки INC файлов
5  .INCLUDE "m8def.inc"           ; загрузка предопределений для ATmega8
6  .LIST                          ; включить генерацию листинга
7
8  .CSEG                          ; начало сегмента кода
9  .ORG 0x0000                    ; начальное значение для адресации
10
11 ; -- инициализация стека --
12 LDI R16, Low(RAMEND) ; младший байт конечного адреса ОЗУ в R16
13 OUT SPL, R16         ; установка младшего байта указателя стека
14 LDI R16, High(RAMEND) ; старший байт конечного адреса ОЗУ в R16
15 OUT SPH, R16         ; установка старшего байта указателя стека
16
17 .equ Delay = 5        ; установка константы времени задержки
18
19 ; -- устанавливаем каналы PD0 и PD1 порта PORTD (PD) на вывод --
20 LDI R16, 0b0000011    ; поместим в регистр R16 число 3 (0x3)
21 OUT DDRD, R16         ; загрузим значение из регистра R16 в порт DDRD
22

```

Рисунок 6.3 – Частина 1 програми для схемі рис. 6.2.

```

22
23 ; -- основной цикл программы --
24 Start:
25     SBI PORTD, PORTD0 ; подача на пин с каналом PD0 высокого уровня
26     CBI PORTD, PORTD1 ; подача на пин с каналом PD1 низкого уровня
27     RCALL Wait        ; вызываем подпрограмму задержки по времени
28     SBI PORTD, PORTD1 ; подача на пин с каналом PD1 высокого уровня
29     CBI PORTD, PORTD0
30     RCALL Wait
31     RJMP Start        ; возврат к метке Start, повторяем все в цикле
32
33 ; -- подпрограмма задержки по времени --
34 Wait:
35     LDI R17, Delay    ; загрузка константы для задержки в регистр R17
36 WLoop0:
37     LDI R18, 50        ; загружаем число 50 (0x32) в регистр R18
38 WLoop1:
39     LDI R19, 0xC8      ; загружаем число 200 (0xC8, $C8) в регистр R19
40 WLoop2:
41     DEC R19            ; уменьшаем значение в регистре R19 на 1
42     BRNE WLoop2        ; возврат к WLoop2 если значение в R19 не равно 0
43     DEC R18            ; уменьшаем значение в регистре R18 на 1
44     BRNE WLoop1        ; возврат к WLoop1 если значение в R18 не равно 0
45     DEC R17            ; уменьшаем значение в регистре R17 на 1
46     BRNE WLoop0        ; возврат к WLoop0 если значение в R17 не равно 0
47     RET                ; возврат из подпрограммы Wait
48

```

Рисунок 6.4 – Частина 2 програми для схеми рис. 6.2.

8. Модифікувати програму у частині зміни частоти або порядку миготіння, кількості світлодіодів.

Контрольні питання

1. Які функції виконують елементи схеми рис. 6.1 та 6.2.
2. Призначення інструкцій контролера I8051 у програмі, що керує схемою рис. 6.1.
3. Призначення інструкцій контролера AVR, що є у програмі наведеної на рис. 6.1.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Програмування мікроконтролерів AVR : [навчальний посібник] / С. М. Цирульник, О. Д. Азаров, Л. В. Крупельницький, Т. І. Трояновська. – Вінниця : ВНТУ, 2018. – 111 с. [Tsirulnik_2018_111.pdf \(vntu.edu.ua\)](https://vntu.edu.ua/Tsirulnik_2018_111.pdf)

2. Навчальний посібник з дисципліни «Проектування мікропроцесорних систем», розділ «Програмування мікроконтролерів родини AVR» для студентів напряму підготовки 6.050201 «Системна інженерія» кафедри Автоматики та управління у технічних системах / Укл.: А.О. Новацький – К: НТУУ „КПІ”, 2013 – 109 с.

[ПОСІБНИК-ПРОГРАМУВАННЯ-AVR.pdf \(kpi.ua\)](https://kpi.ua/posibnik-programuvannya-avr.pdf)

3. Міліх В. І. Електротехніка, електроніка та мікропроцесорна техніка [Текст]: підручник для студ. вищих навч. закладів / В. І. Міліх, О. О. Шавьолкін ; за ред. В. І. Міліх. - 3-ге вид. – Київ : Каравела, 2016. – 688 с.

4. Мікропроцесорні та мікроконтролерні системи : Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. освітньої програми «Інтегровані інформаційні системи» спеціальності 126 «Інформаційні системи та технології», / КПІ ім. Ігоря Сікорського; уклад.: А.О. Новацький. – Електронні текстові дані (1 файл: 18.983 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 247 с

5. Розробка радіоелектронних схем на основі мікроконтролерів (на прикладі AVR мікроконтролерів фірми Atmel): методичний посібник до курсу "Проектування радіоелектронних схем" для студентів радіофізичного факультету / Пархоменко Д. А., Смирнов Є. М. – Київ: Радіофізичний факультет Київського національного університету імені Тараса Шевченка, 2013. – 74 с.

Допоміжна

Інформаційні ресурси

6. Intel MCS-51 [Електронний ресурс]. Режим доступу [https://uk.wikipedia.org/wiki/Intel MCS-51](https://uk.wikipedia.org/wiki/Intel_MCS-51) (дата звернення: 23.12.2023).

7. Грищук Ю.С. Мікропроцесорні пристрої: навчальний посібник. – Харків: НТУ «ХПІ», 2005.- 280с. Електронний документ. Режим доступу <http://web.kpi.kharkov.ua/ea/wp-content/uploads/sites/25/2013/04/Mikroprotsesorni-pristroyi.pdf> (дата звернення: 23.12.2023).

8. Мікропроцесорна техніка: Практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 163 «Біомедична інженерія» та 152 «Метрологія та інформаційно-вимірвальна техніка»/ В.В. Шликов; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 3,1 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 144 с.. Режим доступу http://ela.kpi.ua/bitstream/123456789/24694/3/Shlykov_microprotsessor_tekhn%D0%BA%D0%B0_praktykum.pdf (дата звернення: 23.12.2023).

9. Офіційний сайт проекту Arduino / [Електронний ресурс] – Режим доступу: <https://www.arduino.cc/>. (дата звернення: 23.12.2023).

10. Довідник мови Ардуїно / [Електронний ресурс]. – Режим доступу: <https://doc.arduino.ua/ru/prog/> . (дата звернення: 23.12.2023).

11. Офіційний сайт UnoArduSim [Електронний ресурс] <https://www.sites.google.com/site/unoardusim/services>. (дата звернення: 23.12.2023).

12. Офіційний сайт EdSim51. **EdSim51 - The 8051 Simulator for Teachers and Students** [Електронний ресурс]/ - Режим доступу: <https://www.edsim51.com> (дата звернення: 23.12.2023).

13. Офіційний сайт Tinkercad [Електронний ресурс] - Режим доступу: <https://www.tinkercad.com/> (дата звернення: 23.12.2023).

14. Офіційний сайт Proteus [Електронний ресурс] – Режим доступу: <http://www.labcenter-electronics.com/> (дата звернення: 23.12.2023).

15. Сайт Electronic Components Datasheet Search [Електронний ресурс] – Режим доступу: <https://www.alldatasheet.com/> (дата звернення: 23.12.2023).